Efficient Graph-Based Semi-Supervised Learning of Structured Tagging Models

Amarnag Subramanya
Google Research
Mountain View, CA 94043
asubram@google.com

Slav Petrov
Google Research
New York, NY 10011
slav@google.com

Fernando Pereira
Google Research
Mountain View, CA 94043
pereira@google.com

Abstract

We describe a new scalable algorithm for semi-supervised training of conditional random fields (CRF) and its application to part-of-speech (POS) tagging. The algorithm uses a similarity graph to encourage similar n-grams to have similar POS tags. We demonstrate the efficacy of our approach on a domain adaptation task, where we assume that we have access to large amounts of unlabeled data from the target domain, but no additional labeled data. The similarity graph is used during training to smooth the state posteriors on the target domain. Standard inference can be used at test time. Our approach is able to scale to very large problems and yields significantly improved target domain accuracy.

1 Introduction

Semi-supervised learning (SSL) is the use of small amounts of labeled data with relatively large amounts of unlabeled data to train predictors. In some cases, the labeled data can be sufficient to provide reasonable accuracy on in-domain data, but performance on even closely related out-of-domain data may lag far behind. Annotating training data for all sub-domains of a varied domain such as all of Web text is impractical, giving impetus to the development of SSL techniques that can learn from unlabeled data to perform well across domains. The earliest SSL algorithm is self-training (Scudder, 1965), where one makes use of a previously trained model to annotate unlabeled data which is then used to re-train the model. While self-training is widely used and can yield good results in some applications (Yarowsky, 1995), it has no theoretical guarantees except under certain stringent conditions, which rarely hold in practice(Haffari and Sarkar, 2007).

Other SSL methods include co-training (Blum and Mitchell, 1998), transductive support vector machines (SVMs) (Joachims, 1999), and graph-based SSL (Zhu et al., 2003). Several surveys cover a broad range of methods (Seeger, 2000; Zhu, 2005; Chapelle et al., 2007; Blitzer and Zhu, 2008). A majority of SSL algorithms are computationally expensive; for example, solving a transductive SVM exactly is intractable. Thus we have a conflict between wanting to use SSL with large unlabeled data sets for best accuracy, but being unable to do so because of computational complexity. Some researchers attempted to resolve this conflict by resorting to approximations (Collobert et al., 2006), but those lead to suboptimal results (Chapelle et al., 2007).

Graph-based SSL algorithms (Zhu et al., 2003; Joachims, 2003; Corduneanu and Jaakkola, 2003; Belkin et al., 2005; Subramanya and Bilmes, 2009) are an important subclass of SSL techniques that have received much attention in the recent past, as they outperform other approaches and also scale easily to large problems. Here one assumes that the data (both labeled and unlabeled) is represented by vertices in a graph. Graph edges link vertices that are likely to have the same label. Edge weights govern how strongly the labels of the nodes linked by the edge should agree.

Most previous work in SSL has focused on unstructured classification problems, that is, problems with a relatively small set of atomic labels. There has been much less work on SSL for structured prediction where labels are composites of many atomic labels with constraints between them. While the number of atomic labels might be small, there will generally be exponentially many ways to combine them into the final structured label. Structured prediction problems over sequences appear for example in speech recognition, named-entity recognition, and part-of-speech tagging; in machine translation and syntactic parsing, the output may be tree-structured.

Altun et al. (2005) proposed a max-margin objective for semi-supervised learning over structured spaces. Their objective is similar to that of manifold regularization (Belkin et al., 2005) and they make use of a graph as a smoothness regularizer. However their solution involves inverting a matrix whose size depends on problem size, making it impractical for very large problems. Brefeld and Scheffer (2006) present a modified version of the co-training algorithm for structured output spaces. In both of the above cases, the underlying model is based on structured SVM, which does not scale well to very large datasets. More recently Wang et al. (2009) proposed to train a conditional random field (CRF) (Lafferty et al., 2001) using an entropy-based regularizer. Their approach is similar to the entropy minimization algorithm (Grandvalet and Bengio, 2005). The problem here is that their objective is not convex and thus can pose issues for large problems. Further, graph-based SSL algorithms outperform algorithms based on entropy minimization (Chapelle et al., 2007).

In this work, we propose a graph-based SSL method for CRFs that is computationally practical for very large problems, unlike the methods in the studies cited above. Our method is scalable because it trains with efficient standard building blocks for CRF inference and learning and also standard graph label propagation machinery. Graph regularizer computations are only used for training, so at test time, standard CRF inference can be used, unlike in graph-based transductive methods. Briefly, our approach starts by training a CRF on the source domain labeled data, and then uses it to decode unlabeled data from the target domain. The state posteriors on the target domain are then smoothed using the graph regularizer. Best state sequences for the unlabeled target data are then created by Viterbi decoding with the smoothed state posteriors, and this automatic target domain annotation is combined with the labeled source domain data to retrain the CRF.

We demonstrate our new method in domain adaptation for a CRF part-of-speech (POS) tagger. While POS tagging accuracies have reached the level of inter-annotator agreement (>97%) on the standard PennTreebank test set (Toutanova et al., 2003; Shen et al., 2007), performance on out-of-domain data is often well below 90%, impairing language processing tasks that need syntactic information. For example, on the question domain used in this paper, the tagging accuracy of a supervised CRF is only 84%. Our domain adaptation algorithm improves performance to 87%, which is still far below in-domain performance, but a significant reduction in error.

2 Supervised CRF

We assume that we have a set of labeled source domain examples $\mathcal{D}_l = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, but only unlabeled target domain examples $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$. Here $\mathbf{x}_i = x_i^{(1)} x_i^{(2)} \cdots x_i^{(|\mathbf{x}_i|)}$ is the sequence of words in sentence i and $\mathbf{y}_i = y_i^{(1)} y_i^{(2)} \cdots y_i^{(|\mathbf{x}_i|)}$ is the corresponding POS tag sequence, with $y_i^{(j)} \in \mathcal{Y}$ where $\mathcal{Y}$ is the set of POS tags. Our goal is to learn a CRF of the form:

$$\mathrm{p}(\mathbf{y}_i|\mathbf{x}_i;\Lambda) \propto \exp\left(\sum_{j=1}^{N_i}\sum_{k=1}^{K}\lambda_k f_k(y_i^{(j-1)}, y_i^{(j)}, \mathbf{x}_i, j)\right)$$

for the target domain. In the above equation, $\Lambda = \{\lambda_1, \ldots, \lambda_K\} \in \mathbb{R}^K$, $f_k(y_i^{(j-1)}, y_i^{(j)}, \mathbf{x}_i, j)$ is the k-th feature function applied to two consecutive CRF states and some window of the input sequence, and λ_k is the weight of that feature. We discuss our features in detail in Section 6. Given only labeled data $\mathcal{D}_l$, the optimal feature weights are given by:

$$\Lambda^* = \underset{\Lambda \in \mathbb{R}^K}{\operatorname{argmin}} \left[-\sum_{i=1}^{l} \log \mathrm{p}(\mathbf{y}_i|\mathbf{x}_i;\Lambda) + \gamma\|\Lambda\|^2 \right] \quad (1)$$

Here $\|\Lambda\|^2$ is the squared ℓ_2-norm and acts as the regularizer, and γ is a trade-off parameter whose setting we discuss in Section 6. In our case, we also have access to the unlabeled data $\mathcal{D}_u$ from the target domain which we would like to use for training the CRF. We first describe how we construct a similarity graph over the unlabeled which will be used in our algorithm as a graph regularizer.

3 Graph Construction

Graph construction is the most important step in graph-based SSL. The standard approach for unstructured problems is to construct a graph whose vertices are labeled and unlabeled examples, and whose weighted edges encode the degree to which the examples they link should have the same label (Zhu et al., 2003). Then the main graph construction choice is what similarity function to use for the weighted edges between examples. However, in structured problems the situation is more complicated. Consider the case of sequence tagging we are studying. While we might be able to choose some appropriate sequence similarity to construct the graph, such as edit distance or a string kernel, it is not clear how to use whole sequence similarity to constrain whole tag sequences assigned to linked examples in the learning algorithm. Altun et al. (2005) had the nice insight of doing the graph construction not for complete structured examples but instead for the *parts* of structured examples (also known as factors in graphical model terminology), which encode the local dependencies between input data and output labels in the structured problem. However, their approach is too demanding computationally (see Section 5), so instead we use local sequence contexts as graph vertices, exploting the empirical observation that the part of speech of a word occurrence is mostly determined by its local context.

Specifically, the set V of graph vertices consists of all the word n-grams[1] (*types*) that have occurrences (*tokens*) in training sentences (labeled and unlabeled). We partition $V = V_l \cup V_u$ where V_l corresponds to n-grams that occur at least once in the labeled data, and V_u corresponds to n-grams that occur only in the unlabeled data.

Given a symmetric similarity function between types to be defined below, we link types u and v with

[1]We pad the n-grams at the beginning and end of sentences with appropriate dummy symbols.

Description	Feature
Trigram + Context	$x_1\ x_2\ x_3\ x_4\ x_5$
Trigram	$x_2\ x_3\ x_4$
Left Context	$x_1\ x_2$
Right Context	$x_4\ x_5$
Center Word	x_2
Trigram – Center Word	$x_2\ x_4$
Left Word + Right Context	$x_2\ x_4\ x_5$
Left Context + Right Word	$x_1\ x_2\ x_4$
Suffix	HasSuffix(x_3)

Table 1: Features we extract given a sequence of words "$x_1\ x_2\ x_3\ x_4\ x_5$" where the trigram is "$x_2\ x_3\ x_4$".

an edge of weight w_{uv}, defined as:

$$w_{uv} = \begin{cases} \mathrm{sim}(u, v) & \text{if } v \in \mathcal{K}(u) \text{ or } u \in \mathcal{K}(v) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{K}(u)$ is the set of k-nearest neighbors of u according to the given similarity. For all experiments in this paper, $n = 3$ and $k = 5$.

To define the similarity function, for each token of a given type in the labeled and unlabeled data, we extract a set of context features. For example, for the token $x_2\ x_3\ x_4$ occurring in the sequence $x_1\ x_2\ x_3\ x_4\ x_5$, we use feature templates that capture the left ($x_1\ x_2$) and right contexts ($x_4\ x_5$). Additionally, we extract suffix features from the word in the middle. Table 1 gives an overview of the features that we used. For each n-gram type, we compute the vector of pointwise mutual information (PMI) values between the type and each of the features that occur with tokens of that type. Finally, we use the cosine distance between those PMI vectors as our similarity function.

We have thus circumvented the problem of defining similarities over sequences by defining the graph over types that represent local sequence contexts. Since our CRF tagger only uses local features of the input to score tag pairs, we believe that the graph we construct captures all significant context information. Figure 1 shows an excerpt from our graph. The figure shows the neighborhoods of a subset of the vertices with the center word 'book.' To reduce clutter, we included only closest neighbors and the edges that involve the nodes of interest.

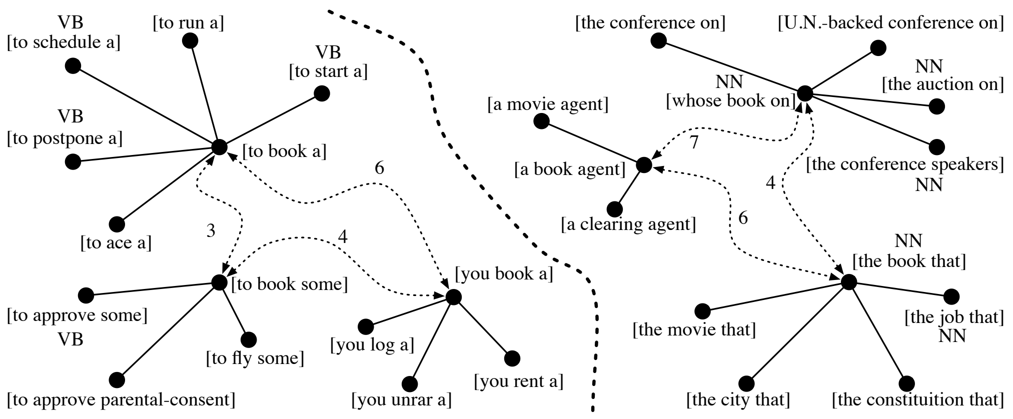

Figure 1: Vertices with center word 'book' and their local neighborhoods, as well as the shortest-path distance between them. Note that the noun (NN) and verb (VB) interpretations form two disjoint connected components.

It is remarkable that the neighborhoods are coherent, showing very similar syntactic configurations. Furthermore, different vertices that (should) have the same label are close to each other, forming connected components for each part-of-speech category (for nouns and verbs in the figure). We expect the similarity graph to provide information that cannot be expressed directly in a sequence model. In particular, it is not possible in a CRF to directly enforce the constraint that similar trigrams appearing in different sentences should have similar POS tags. This constraint however is important during (semi-supervised) learning, and is what makes our approach different and more effective than self-training.

In practice, we expect two main benefits from our graph-based approach. First, the graph allows new features to be discovered. Many words occur only in the unlabeled data and a purely supervised CRF would not be able to learn feature weights for those observations. We could use self-training to learn weights for those features, but self-training just tends to reinforce the knowledge that the supervised model already has. The similarity graph on the other hand can link events that occur only in the unlabeled data to similar events in the labeled data. Furthermore, because the graph is built over types rather than tokens, it will encourage the same interpretation to be chosen for similar trigrams occurring in different sentences. For example, the word 'unrar' will most likely not occur in the labeled training data. Seeing it in the neighborhood of words for which we know the POS tag will help us learn the correct POS tag for this otherwise unknown word (see Figure 1).

Second, the graph propagates adjustments to the weights of known features. Many words occur only a handful of times in our labeled data, resulting in poor estimates of their contributions. Even for frequently occurring events, their distribution in the target domain might be different from their distribution in the source domain. While self-training might be able to help adapt to such domain changes, its effectiveness will be limited because the model will always be inherently biased towards the source domain. In contrast, labeled vertices in the similarity graph can help disambiguate ambiguous contexts and correct (some of) the errors of the supervised model.

4 Semi-Supervised CRF

Given unlabeled data $\mathcal{D}_u$, we only have access to the prior $p(\mathbf{x})$. As the CRF is a discriminative model, the lack of label information renders the CRF weights independent of $p(\mathbf{x})$ and thus we cannot directly utilize the unlabeled data when training the CRF. Therefore, semi-supervised approaches to training discriminative models typically use the unlabeled data to construct a regularizer that is used to guide the learning process (Joachims, 1999; Lawrence and Jordan, 2005). Here we use the graph as a smoothness regularizer to train CRFs in a semi-supervised manner.

Our algorithm iterates between the following five

Algorithm 1 Semi-Supervised CRF Training

```
$\Lambda^s$ = crf-train($\mathcal{D}_l$, $\Lambda^0$)
Set $\Lambda_0^{(t)} = \Lambda^{(s)}$
**while** not converged **do**
    $\{\mathrm{p}\}$ = posterior_decode($\mathcal{D}_u$, $\Lambda_{old}$)
    $\{\mathrm{q}\}$ = token_to_type($\{\mathrm{p}\}$)
    $\{\hat{\mathrm{q}}\}$ = graph_propagate($\{\mathrm{q}\}$)
    $\mathcal{D}_u^{(1)}$ = viterbi_decode($\{\hat{\mathrm{q}}\}$, $\Lambda_{old}$)
    $\Lambda_{n+1}^{(t)}$ = crf-train($\mathcal{D}_l \cup \mathcal{D}_u^{(1)}$, $\Lambda_n^{(t)}$)
**end while**
Return last $\Lambda^{(t)}$
```

simple (and convex) steps: Given a set of CRF parameters, we first compute marginals over the unlabeled data (posterior_decode). The marginals over tokens are then aggregated to marginals over types (token_to_type), which are used to initialize the graph label distributions. After running label propagation (graph_propagate), the posteriors from the graph are used to smooth the state posteriors. Decoding the unlabeled data (viterbi_decode) produces a new set of automatic annotations that can be combined with the labeled data to retrain the CRF using the supervised CRF training objective (crf-train). These steps, summarized in Algorithm 1, are iterated until convergence.

4.1 Posterior Decoding

Let $\Lambda_n^{(t)}$ (t refers to target domain) represent the estimate of the CRF parameters for the target domain after the n-th iteration.[2] In this step, we use the current parameter estimates to compute the marginal probabilities

$$\mathrm{p}(y_i^{(j)}|\mathbf{x}_i;\Lambda_n^{(t)})\ 1 \le j \le |\mathbf{x}_i|, i \in \mathcal{D}_l$$

over POS tags for every word position j for i indexing over sentences in $\mathcal{D}_l \cup \mathcal{D}_u$.

4.2 Token-to-Type Mapping

Recall that our graph is defined over types while the posteriors computed above involve particular tokens. We accumulate token-based marginals to create type marginals as follows. For a sentence i and word position j in that sentence, let $T(i,j)$ be the trigram (graph node) centered at position j. Conversely, for a trigram type u, let $T^{-1}(u)$ be the set of actual occurrences (tokens) of that trigram u; that is, all pairs (i,j) where i is the index of a sentence where u occurs and j is the position of the center word of an occurrence of u in that sentence. We calculate type-level posteriors as follows:

$$\mathrm{q}_u(y) \triangleq \frac{1}{|T^{-1}(u)|} \sum_{(i,j)\in T^{-1}(u)} \mathrm{p}(y_i^{(j)}|\mathbf{x}_i;\Lambda_n^{(t)}) \quad .$$

This combination rule connects the token-centered CRF with the type-centered graph. Other ways of combining the token marginals, such as using weights derived from the entropies of marginals, might be worth investigating.

4.3 Graph Propagation

We now use our similarity graph (Section 3) to smooth the type-level marginals by minimizing the following convex objective:

$$\begin{aligned} \mathcal{C}(\mathrm{q}) &= \sum_{u\in V_l} \|\mathrm{r}_u - \mathrm{q}_u\|^2 \\ &+ \mu \sum_{u\in V, v\in\mathcal{N}(i)} w_{uv}\|\mathrm{q}_u - \mathrm{q}_v\|^2 + \nu \sum_{u\in V}\|\mathrm{q}_u - U\|^2 \\ \text{s.t.} &\sum_y \mathrm{q}_u(y) = 1\ \forall u\ \&\ \mathrm{q}_u(y) \ge 0\ \forall u, y \end{aligned} \qquad (2)$$

where $\mathrm{q} = \{\mathrm{q}_1, \mathrm{q}_2, \dots \mathrm{q}_{|V|}\}$. The setting of the hyperparameters μ and ν will be discussed in Section 6, $\mathcal{N}(u)$ is the set of neighbors of node u, and r_u is the empirical marginal label distribution for trigram u in the labeled data. We use a squared loss to penalize neighboring nodes that have different label distributions: $\|\mathrm{q}_u - \mathrm{q}_v\|^2 = \sum_y(\mathrm{q}_u(y) - \mathrm{q}_v(y))^2$, additionally regularizing the label distributions towards the uniform distribution U over all possible labels $\mathcal{Y}$. It can be shown that the above objective is convex in q.

Our graph propagation objective can be seen as a multi-class generalization of the quadratic cost criterion (Bengio et al., 2007). The first term in the above objective requires that we respect the information in our labeled data. The second term is the graph smoothness regularizer which requires that the q_i's be smooth with respect to the graph. In other words, if w_{uv} is large, then q_u and q_v should be close in the

[2]In the first iteration, we initialize the target domain parameters to the source domain parameters: $\Lambda_0^{(t)} = \Lambda^{(s)}$.

squared-error sense. This implies that vertices u and v are likely to have similar marginals over POS tags. The last term is a regularizer and encourages all type marginals to be uniform to the extent that is allowed by the first two terms. If a unlabeled vertex does not have a path to any labeled vertex, this term ensures that the converged marginal for this vertex will be uniform over all tags, ensuring that our algorithm performs at least as well as a standard self-training based algorithm, as we will see later.

While the objective in Equation 2 admits a closed form solution, it involves inverting a matrix of order $|V|$ and thus we use instead the simple iterative update given by

$$\begin{aligned} q_u^{(m)}(y) &= \frac{\gamma_u(y)}{\kappa_u} \text{ where} \\ \gamma_u(y) &= r_u(y)\delta(u \in V_l) \\ &\quad + \sum_{v \in \mathcal{N}(u)} w_{uv} q_v^{(m-1)}(y) + \nu U(y), \\ \kappa_u &= \delta(u \in V_l) + \nu + \mu \sum_{v \in \mathcal{N}(u)} w_{uv} \end{aligned} \tag{3}$$

where m is the iteration index and δ is the indicator function that returns 1 if and only if the condition is true. The iterative procedure starts with $\mathrm{q}_u^{(0)}(y) = \mathrm{q}_u(y)$ as given in the previous section. In all our experiments we run 10 iterations of the above algorithm, and we denote the type marginals at completion by $\mathrm{q}_u^*(y)$.

4.4 Viterbi Decoding

Given the type marginals computed in the previous step, we interpolate them with the original CRF token marginals. This interpolation between type and token marginals encourages similar n-grams to have similar posteriors, while still allowing n-grams in different sentences to differ in their posteriors. For each unlabeled sentence i and word position j in it, we calculate the following interpolated tag marginal:

$$\begin{aligned} \hat{\mathrm{p}}(y_i^{(j)} = y|\mathbf{x}_i) &= \alpha \mathrm{p}(y_i^{(j)} = y|\mathbf{x}_i; \Lambda_n^{(t)}) \\ &\quad + (1-\alpha)\mathrm{q}^*_{T(m,n)}(y) \end{aligned} \tag{4}$$

where α is a mixing coefficient which reflects the relative confidence between the original posteriors from the CRF and the smoothed posteriors from the graph. We discuss how we set α in Section 6.

The interpolated marginals summarize all the information obtained so far about the tag distribution at each position. However, if we were to use them on their own to select the most likely POS tag sequence, the first-order tag dependencies modeled by the CRF would be mostly ignored. This happens because the type marginals obtained from the graph after label propagation will have lost most of the sequence information. To enforce the first-order tag dependencies we therefore use Viterbi decoding over the combined interpolated marginals and the CRF transition potentials to compute the best POS tag sequence for each unlabeled sentence. We refer to these 1-best transcripts as $\mathbf{y}_i^*,\ i \in \mathcal{D}_u$.

4.5 Re-training the CRF

Now that we have successfully labeled the unlabeled target domain data, we can use it in conjunction with the source domain labeled data to re-train the CRF:

$$\begin{aligned} \Lambda_{n+1}^{(t)} = &\underset{\Lambda \in \mathbb{R}^K}{\operatorname{argmin}} \Bigg[-\sum_{i=1}^{l} \log \mathrm{p}(\mathbf{y}_i|\mathbf{x}_i; \Lambda_n^{(t)}) \\ &- \eta \sum_{i=l+1}^{l+u} \log \mathrm{p}(\mathbf{y}_i^*|\mathbf{x}_i; \Lambda_n^{(t)}) + \gamma \|\Lambda\|^2 \Bigg] \end{aligned} \tag{5}$$

where η and γ are hyper-parameters whose setting we discuss in Section 6. Given the new CRF parameters Λ we loop back to step 1 (Section 4.1) and iterate until convergence. It is important to note that every step of our algorithm is convex, although their combination clearly is not.

5 Related Work

Our work differs from previous studies of SSL (Blitzer et al., 2006; III, 2007; Huang and Yates, 2009) for improving POS tagging in several ways. First, our algorithm can be generalized to other structured semi-supervised learning problems, although POS tagging is our motivating task and test application. Unlike III (2007), we do not require target domain labeled data. While the SCL algorithm (Blitzer et al., 2006) has been evaluated without target domain labeled data, that evaluation was to some extent transductive in that the target test data (unlabeled) was included in the unsupervised stage of SCL training that creates the structural correspondence between the two domains.

We mentioned already the algorithm of Altun et al. (2005), which is unlikely to scale up because its dual formulation requires the inversion of a matrix whose size depends on the graph size. Gupta et al. (2009) also constrain similar trigrams to have similar POS tags by forming cliques of similar trigrams and maximizing the agreement score over these cliques. Computing clique agreement potentials however is NP-hard and so they propose approximation algorithms that are still quite complex computationally. We achieve similar effects by using our simple, scalable convex graph regularization framework. Further, unlike other graph-propagation algorithms (Alexandrescu and Kirchhoff, 2009), our approach is inductive. While one might be able to make inductive extensions of transductive approaches (Sindhwani et al., 2005), these usually require extensive computational resources at test time.

6 Experiments and Results

We use the Wall Street Journal (WSJ) section of the Penn Treebank as our labeled source domain training set. We follow standard setup procedures for this task and train on sections 00-18, comprising of 38,219 POS-tagged sentences with a total of 912,344 words. To evaluate our domain-adaptation approach, we consider two different target domains: questions and biomedical data. Both target domains are relatively far from the source domain (newswire), making this a very challenging task.

The QuestionBank (Judge et al., 2006), provides an excellent corpus consisting of 4,000 questions that were manually annotated with POS tags and parse trees. We used the first half as our development set and the second half as our test set. Questions are difficult to tag with WSJ-trained taggers primarily because the word order is very different than that of the mostly declarative sentences in the training data. Additionally, the unknown word rate is more than twice as high as on the in-domain development set (7.29% vs. 3.39%). As our unlabeled data, we use a set of 10 million questions collected from anonymized Internet search queries. These queries were selected to be similar in style and length to the questions in the QuestionBank.[3] As running the CRF over 10 million sentences can be rather cumbersome and probably unnecessary, we randomly select 100,000 of these queries and treat this as $\mathcal{D}_u$. Because the graph nodes and the features used in the similarity function are based on n-grams, data sparsity can be a serious problem, and we therefore use the entire unlabeled data set for graph construction. We estimate the mutual information-based features for each trigram type over all the 10 million questions, and then construct the graph over only the set of trigram types that actually occurs in the 100,000 random subset and the WSJ training set.

For our second target domain, we use the Penn BioTreebank (PennBioIE, 2005). This corpus consists of 1,061 sentences that have been manually annotated with POS tags. We used the first 500 sentences as a development set and the remaining 561 sentences as our final test set. The high unknown word rate (23.27%) makes this corpus very difficult to tag. Furthermore, the POS tag set for this data is a super-set of the Penn Treebank's, including the two new tags HYPH (for hyphens) and AFX (for common post-modifiers of biomedical entities such as genes). These tags were introduced due to the importance of hyphenated entities in biomedical text, and are used for 1.8% of the words in the test set. Any tagger trained only on WSJ text will automatically predict wrong tags for those words. For unlabeled data we used 100,000 sentences that were chosen by searching MEDLINE for abstracts pertaining to cancer, in particular genomic variations and mutations (Blitzer et al., 2006). Since we did not have access to additional unlabeled data, we used the same set of sentences as target domain unlabeled data, $\mathcal{D}_u$. The graph here was constructed over the 100,000 unlabeled sentences and the WSJ training set. Finally, we remind the reader that we did not use label information for graph construction in either corpus.

6.1 Baselines

Our baseline supervised CRF is competitive with state-of-the-art discriminative POS taggers (Toutanova et al., 2003; Shen et al., 2007), achieving 97.17% on the WSJ development set (sections 19-21). We use a fairly standard set of features, including word identity, suffixes and prefixes and detectors

[3]In particular, we selected queries that start with an English function word that can be used to start a question (what, who, when, etc.), and have between 30 and 160 characters.

	Questions		Bio	
	Dev	Eval	Dev	Eval
Supervised CRF	84.8	83.8	86.5	86.2
Self-trained CRF	85.4	84.0	87.5	87.1
Semi-supervised CRF	**87.6**	**86.8**	**87.5**	**87.6**

Table 2: Domain adaptation experiments. POS tagging accuracies in %.

for special characters such as dashes and digits. We do not use of observation-dependent transition features. Both supervised and semi-supervised models are regularized with a squared ℓ_2-norm regularizer with weight set to 0.01.

In addition to the supervised baseline trained exclusively on the WSJ, we also consider a semi-supervised self-trained baseline ("Self-trained CRF" in Table 2). In this approach, we first train a supervised CRF on the labeled data and then do semi-supervised training without label propagation. This is different from plain self-training because it aggregates the posteriors over tokens into posteriors over types. This aggregation step allows instances of the same trigram in different sentences to share information and works better in practice than direct self-training on the output of the supervised CRF.

6.2 Domain Adaptation Results

The data set obtained concatenating the WSJ training set with the 10 million questions had about 20 million trigram types. Of those, only about 1.1 million trigram types occurred in the WSJ training set or in the 100,000 sentence sub-sample. For the biomedical domain, the graph had about 2.2 million trigrams. For all our experiments we set hyperparameters as follows: for graph propagation, $\mu = 0.5$, $\nu = 0.01$, for Viterbi decoding mixing, $\alpha = 0.6$, for CRF re-training, $\eta = 0.001$, $\gamma = 0.01$. These parameters were chosen based on development set performance. All CRF objectives were optimized using L-BFGS (Bertsekas, 2004).

Table 2 shows the results for both domains. For the question corpus, the supervised CRF performs at only 85% on the development set. While it is almost impossible to improve in-domain tagging accuracy and tagging is therefore considered a solved problem by many, these results clearly show that the problem is far from solved. Self-training improves over the baseline by about 0.6% on the development set. However the gains from self-training are more modest (0.2%) on the evaluation (test) set. Our approach is able to provide a more solid improvement of about 3% absolute over the supervised baseline and about 2% absolute over the self-trained system on the question development set. Unlike self-training, on the question evaluation set, our approach provides about 3% absolute improvement over the supervised baseline. For the biomedical data, while the performances of our approach and self-training are statistically indistinguishable on the development set, we see modest gains of about 0.5% absolute on the evaluation set. On the same data, we see that our approach provides about 1.4% absolute improvement over the supervised baseline.

7 Analysis & Conclusion

The results suggest that our proposed approach provides higher gains relative to self-training on the question data than on the biomedical corpus. We hypothesize that this caused by sparsity in the graph generated from the biomedical dataset. For the questions graph, the PMI statistics were estimated over 10 million sentences while in the case of the biomedical dataset, the same statistics were computed over just 100,000 sentences. We hypothesize that the lack of well-estimated features in the case of the biomedical dataset leads to a sparse graph.

To verify the above hypothesis, we measured the percentage of trigrams that occur in the target domain (unlabeled) data that do not have any path to a trigram in the source domain data, and the average minimum path length between a trigram in the target data and a trigram in the source data (when such a path exists). The results are shown in Table 3. For the biomedical data, close to 50% of the trigrams from the target data do not have a path to a trigram from the source data. Even when such a path exists, the average path length is about 22. On

	Questions	Bio
% of unlabeled trigrams not connected to any labeled trigrams	12.4	46.8
average path length between an unlabeled trigram and its nearest labeled trigram	9.4	22.4

Table 3: Analysis of the graphs constructed for the two datasets discussed in Section 6. Unlabeled trigrams occur in the target domain only. Labeled trigrams occur at least once in the WSJ training data.

the other hand, for the question corpus, only about 12% of the target domain trigrams are disconnected, and the average path length is about 9. These results clearly show the sparse nature of the biomedical graph. We believe that it is this sparsity that causes the graph propagation to *not* have a more noticeable effect on the final performance. It is noteworthy that making use of even such a sparse graph does not lead to any degradation in results, which we attribute to the choice of graph-propagation regularizer (Section 4.3).

We presented a simple, scalable algorithm for training structured prediction models in a semi-supervised manner. The approach is based on using as a regularizer a nearest-neighbor graph constructed over trigram types. Our results show that the approach not only scales to large datasets but also produces significantly improved tagging accuracies.

References

A. Alexandrescu and K. Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *NAACL*.

Y. Altun, D. McAllester, and M. Belkin. 2005. Maximum margin semi-supervised learning for structured variables. In *Advances in Neural Information Processing Systems 18*, page 18.

M. Belkin, P. Niyogi, and V. Sindhwani. 2005. On manifold regularization. In *Proc. of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

Y. Bengio, O. Delalleau, and N. L. Roux, 2007. *Semi-Supervised Learning*, chapter Label Propogation and Quadratic Criterion. MIT Press.

D Bertsekas. 2004. *Nonlinear Programming*. Athena Scientific Publishing.

J. Blitzer and J. Zhu. 2008. ACL 2008 tutorial on Semi-Supervised learning.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP '06*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*.

U. Brefeld and T. Scheffer. 2006. Semi-supervised learning for structured output variables. In *ICML06, 23rd International Conference on Machine Learning*.

O. Chapelle, B. Scholkopf, and A. Zien. 2007. *Semi-Supervised Learning*. MIT Press.

R. Collobert, F. Sinz, J. Weston, L. Bottou, and T. Joachims. 2006. Large scale transductive svms. *Journal of Machine Learning Research*.

A. Corduneanu and T. Jaakkola. 2003. On information regularization. In *Uncertainty in Artificial Intelligence*.

Y. Grandvalet and Y. Bengio. 2005. Semi-supervised learning by entropy minimization. In *CAP*.

R. Gupta, S. Sarawagi, and A. A. Diwan. 2009. Generalized collective inference with symmetric clique potentials. *CoRR*, abs/0907.0589.

G. R. Haffari and A. Sarkar. 2007. Analysis of semi-supervised learning with the Yarowsky algorithm. In *UAI*.

F. Huang and A. Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*. Association for Computational Linguistics.

H. Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

T. Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proc. of the International Conference on Machine Learning (ICML)*.

Thorsten Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proc. of the International Conference on Machine Learning (ICML)*.

J. Judge, A. Cahill, and J. van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguist ics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*.

N. D. Lawrence and M. I. Jordan. 2005. Semi-supervised learning via gaussian processes. In *NIPS*.

PennBioIE. 2005. Mining the bibliome project. In *http://bioie.ldc.upenn.edu/*.

H. J. Scudder. 1965. Probability of Error of some Adaptive Pattern-Recognition Machines. *IEEE Transactions on Information Theory*, 11.

M. Seeger. 2000. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, U.K.

L. Shen, G. Satta, and A. Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL '07*.

V. Sindhwani, P. Niyogi, and M. Belkin. 2005. Beyond the point cloud: from transductive to semi-supervised learning. In *Proc. of the International Conference on Machine Learning (ICML)*.

A. Subramanya and J. A. Bilmes. 2009. Entropic graph regularization in non-parametric semi-supervised classification. In *Neural Information Processing Society (NIPS)*, Vancouver, Canada, December.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL '03*.

Y. Wang, G. Haffari, S. Wang, and G. Mori. 2009. A rate distortion approach for semi-supervised conditional random fields.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the International Conference on Machine Learning (ICML)*.

X. Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.